Docket No.: 122.1487

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
THE BOARD OF PATENT APPEALS AND INTERFERENCES

In re the Application of:

Yoshikazu AOKI

Serial No. 10/073,595 Group Art Unit: 2194

Confirmation No. 4322

Filed: February 12, 2002 Examiner: Andy HO

For: METHOD OF CONTROLLING THE OPERATION OF AN OPERATING SYSTEM IN A COMPUTER SYSTEM

AMENDED APPEAL BRIEF

Commissioner for Patents
PO Box 1450
Alexandria, VA 22313-1450

Attn: **MAIL STOP APPEAL BRIEF-PATENTS**

Sir:

This Amended Appeal Brief is in response to the Notification of Non-Compliant Appeal Brief mailed January 31, 2007, and setting a one-month period for a response, which is set to expire February 28, 2007.

This Amended Appeal Brief is intended to amend the Appeal Brief filed on December 18, 2006 (contents incorporated herein) to include a concise explanation of the subject matter defined in each of the independent claims (claims 1, 16 and 17), by referring to the specification by page and line number and to the drawings. (See amended Sections V and VII). The claimed invention is now mapped to identify independent claims 1, 16 and 17 and dependent claims 4 and 5, by reference to specific page and line numbers of the specification as well as the drawings. (See Section V). Accordingly, it is respectfully submitted that the requirements of 37 CFR 41.37(c)(1)(v) are satisfied.

Further, Section VI is amended to specifically cite the rejections under 35 U.S.C. §103(a), as noted in the aforementioned Notification.

It is respectfully submitted that all requirements under 37 CFR 41.37 are satisfied.

I. REAL PARTY IN INTEREST

The real party in interest in this appeal is the assignee, FUJITSU LIMITED.

II. RELATED APPEALS AND INTERFERENCES

The undersigned attorney, the appellant and the assignee know of no related appeals, interferences or other judicial proceedings which may be related to, directly affect, be directly affected by or have a bearing on the Board's decision in this appeal.

III. STATUS OF THE CLAIMS

Claims 1-18 are currently pending. Claims 1-18 stand finally rejected under 35 U.S.C. §103(a) and claims 1-18 are appealed.

IV. STATUS OF AMENDMENTS

A Response under 37 CFR 1.116 was filed July 31, 2006, in response to the Final Office Action mailed April 18, 2006, but no claim amendments were filed therein. The Response was considered by the Examiner, according to the Advisory Action mailed October 5, 2006.

V. SUMMARY OF CLAIMED SUBJECT MATTER

The present invention relates to a method and a computer-readable storage medium for controlling a computer system by determining and starting a job after recognizing an operation status of the computer system.

Independent claim 1

Independent claim 1, for example, is directed to a method of controlling operation of an operating system in a computer system, according an embodiment of the present invention, which comprises preparing or deleting various kinds of files that show various operation statuses of the computer system in which an operation status represents what process is currently under execution, according to changes in the operation status, and storing the prepared files in a memory section within the computer system. (See, for example, page 3, lines 15-23, page 8, line 31, to page 9, line 1, and page 9, lines 2-5, of the present specification). For example, referring to Fig. 1, at S5 a recovery program is executed, and when the recovery program is

finished, files E and R are deleted at S6. Next, at S11, the execution of program 2 is started (See Fig. 3), and the existence of files E, B and C is checked. In this case these files do not exist, so the backup program is started at S12. When the backup program (S12) has been started, the file B is prepared at S13. When the backup has been a failure, file E is prepared, and file R is prepared when recovery is possible. When the backup has been finished normally, file C is prepared. (See page 8, lines 14-30, and Fig. 1).

As another example, at S21, it is judged that the file C does not exist (in this example) and the flag removal program is executed at S22. Thereafter, files B and C are deleted at S23 (see, for example, page 3, lines 15-23, page 8, line 31, to page 9, line 1, and page 9, lines 2-5, and Fig. 1, reference numerals S21, S22 and S23, of the present application).

A predetermined operation status of the computer system is recognized, depending on whether a file corresponding to the predetermined operation status exists within the memory section or not (see, for example, page 3, lines 23-27, and page 8, lines 14-18, and Fig. 1, reference numeral S11, of the present application). That is, the operation status of the system is recognized based on the presence or absence of files B and C, for example, described above. (See page 9, lines 2-7, of the present specification).

The operation of the operating system is controlled in accordance with a result of the recognition, thereby a job is automatically started, determined based on the recognized operation status, that can be executed in the operation status of the system after the operation status has been recognized (see, for example, page 3, lines 2-7, page 10, lines 25-33, and Fig. 1, reference numerals S3, S12, S21 and S22, of the present application). That is, it is possible to execute a program after changing the operation status.

For example, it is possible to execute the backup program (S12) after executing the recovery program (S5). In other words, referring to Fig. 1, after the recovery program (S5) is run, the existence of files E, B and C are checked. When they do not exist, it is determined, in this case, that the backup program (S12) is to be executed. (See also, page 8, lines 14-18, of the present specification). Similarly, at S21, the existence of file C is checked, and if it judged that file C does not exist, the flag removal program 3 is executed at S22 (see Figs. 1 and 3, and page 8, lines 31-35).

According to the embodiment of the invention described above, it is possible to prepare a mechanism for automatically recognizing the operation status of a computer system, and control the operation of the operating system of the computer system according to the recognized operation status of the system. (See page 12, lines 13-18, of the specification).

Further, according to the present invention, it is possible to provide a method of

automatically starting a job in the computer system that can be executed in an operation status of the system, after this operation status has been recognized. (See page 12, lines 19-25, of the specification).

Independent claim 16

Independent claim 16, for example, is directed to a computer-readable recording medium (see page 11, lines 8-11, of the present specification) that has been recorded with a program for making a computer execute a method of controlling the operation of an operating system in a computer system, the recording medium being recorded with a program comprising preparing or deleting various kinds of files that show various operation statuses of the computer system in which an operation status represents what process is currently under execution, according to changes in the operation status, and storing the prepared files in a memory section within the computer system (see, for example, page 3, lines 15-23, page 8, line 31, to page 9, line 1, and page 9, lines 2-5, and Fig. 1, reference numerals S14, S21 and S23, for example, of the present application). For example, referring to Fig. 1, at S5 a recovery program is executed, and when the recovery program is finished, files E and R are deleted at S6. Next, at S11, the execution of program 2 is started (See Fig. 3), and the existence of files E, B and C is checked. In this case these files do not exist, so the backup program is started at S12. When the backup program (S12) has been started, the file B is prepared at S13. When the backup has been a failure, file E is prepared, and file R is prepared when recovery is possible. When the backup has been finished normally, file C is prepared. (See page 8, lines 14-30, and Fig. 1).

As another example, at S21, it is judged that the file C does not exist (in this example) and the flag removal program is executed at S22. Thereafter, files B and C are deleted at S23 (see, for example, page 3, lines 15-23, page 8, line 31, to page 9, line 1, and page 9, lines 2-5, and Fig. 1, reference numerals S21, S22 and S23, of the present application).

A predetermined operation status of the computer system is recognized, depending on whether a file corresponding to the predetermined operation status exists within the memory section or not (see, for example, page 3, lines 23-27, and page 8, lines 14-18, and Fig. 1, reference numeral S11, of the present application). That is, the operation status of the system is recognized based on the presence or absence of files B and C, for example, described above. (See page 9, lines 2-7, of the present specification).

The operation of the operating system is controlled in accordance with a result of the recognition, thereby a job is automatically started, determined based on the recognized operation status, that can be executed in the operation status of the system after the operation status has been recognized (see, for example, page 3, lines 2-7, page 10, lines 25-33, and Fig.

1, reference numerals S3, S12, S21 and S22, of the present application). That is, it is possible to execute a program after changing the operation status.

For example, it is possible to execute the backup program (S12) after executing the recovery program (S5). In other words, referring to Fig. 1, after the recovery program (S5) is run, the existence of files E, B and C are checked. When they do not exist, it is determined, in this case, that the backup program (S12) is to be executed. (See also, page 8, lines 14-18, of the present specification). Similarly, at S21, the existence of file C is checked, and if it judged that file C does not exist, the flag removal program 3 is executed at S22 (see Figs. 1 and 3, and page 8, lines 31-35).

Independent claim 17

Similarly to independent claim 1, independent claim 17 is directed to a method of controlling the operation of an operating system in a computer system, comprising automatically recognizing an operation status of the computer system in which the operation status represents what process is currently under execution see, for example, page 3, lines 23-27, and page 8, lines 14-18, and Fig. 1, reference numeral S11, of the present application). That is, the operation status of the system is recognized based on the presence or absence of files B and C, for example, described above. (See page 9, lines 2-7, of the present specification).

A job is automatically started, determined based on the recognized operation status (see, for example, page 3, lines 2-7, page 10, lines 25-33, and Fig. 1, reference numerals S3, S12, S21 and S22, of the present application). That is, it is possible to execute a program after changing the operation status. For example, it is possible to execute the backup program (S12) after executing the recovery program (S5). In other words, referring to Fig. 1, after the recovery program (S5) is run, the existence of files E, B and C are checked. When they do not exist, it is determined that the backup program (S12) is to be executed. (See also, page 8, lines 14-18, of the present specification).

Dependent claims 4 and 5

Dependent claims 4 and 5 depend from independent claim 1, but explicitly recite that the starting of the predetermined job is determined based on whether a plurality of the files exist or not within the memory section (see, for example, page 4, lines 3-8, and page 8, lines 16-18, and Fig. 1, reference numeral S12, of the present application). That is, for example, if it is judged that filed E, B and C do not exist, the backup program is executed at S12. (See Fig. 1).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

In the final Office Action, and at issue in this Appeal Brief, claims 1-18 are rejected for obviousness under 35 U.S.C. §103(a) as being unpatentable over Devins et al. (U.S. Patent No. 6,762,761) (hereinafter "Devins").

A key subissue is whether Devins et al. teaches or suggests preparing or deleting various kinds of files that show various operation statuses of the computer system in which an operation status represents what process is currently under execution, according to changes in the operation status, and storing the prepared files in a memory section within the computer system; and controlling the operation of an operating system in accordance with a result of recognizing a predetermined operation status of the computer system, thereby automatically starting a job, determined based on the recognized operation status, that can be executed in the operation status of the system after the operation status has been recognized.

Another key subissue is whether Devins et al. discloses recognizing operation statuses of a computer system, depending on whether a plurality of the files prepared to show various operation statuses of the computer system exists within the memory section.

VII. ARGUMENT

In the final Office Action, the Examiner rejects claims 1-18 for obviousness under 35 U.S.C. §103(a) as being unpatentable over Devins.

References

Devins

In Devins, captured programs are generated and stored in a memory. A graphics processor fetches instructions in a captured program and sends them to a graphics accelerator, which executes the instructions to perform graphics operations. A captured program may include instructions causing the graphics process to monitor the status information in a status register, and delay sending the instructions to the graphics accelerator until specified status information is present. (See Devins, column 2, lines 32-52).

Figs. 2 and 3 of Devins depict a DLP 25 capable of fetching hardware instructions stored in a memory 20, and issuing the instructions to control the accelerator 30. However, the DLP delays the issuance of the instructions in the memory 20 until specified status information is present in status register 100. (See also Devins, column 3, lines 36-67).

Column 6, lines 42-53, of Devins provides an illustrative example in which a program

executes by continuously polling for a triggering event in a hardware status register. When the polling shows that the triggering event has occurred, then the programmed operations are executed. In other words, the programmed operations are predetermined and merely wait for a triggering event before executing. (See also column 8, lines 5-11, of Devins).

Group A: Claims 1-3 and 6-18

Embodiments of the present invention, as recited in independent claim 1, for example, are characterized by preparing or deleting various kinds of files that show various operation statuses of the computer system in which an operation status represents what process is currently under execution, according to changes in the operation status, and storing the prepared files in a memory section within the computer system (see, for example, page 3, lines 15-23, page 8, line 31, to page 9, line 1, and page 9, lines 2-5, and Fig. 1, reference numerals S14, S21 and S23, for example, of the present application); recognizing a predetermined operation status of the computer system, depending on whether a file corresponding to the predetermined operation status exists within the memory section (see, for example, page 3, lines 23-27, and page 8, lines 14-18, and Fig. 1, reference numeral S11, of the present application); and controlling the operation of the operating system in accordance with a result of the recognition, thereby automatically starting a job, determined based on the recognized predetermined operation status, that can be executed in the operation status of the system after the operation status has been recognized (see, for example, page 3, lines 2-7, page 10, lines 25-33, and Fig. 1, reference numerals S3, S12, S21 and S22, of the present application).

On page 3 of the Action mailed October 31, 2005, the Examiner cites Devins, column 3, line 58, to column 4, line 7, as disclosing automatically starting a job, determined based on the recognized predetermined operation status. However, the cited portion of Devins discloses a device driver 15 including a capture routine 50 and I/O directives causing the DLP 25 to monitor the status of register 100 in the accelerator 30. The device driver 15 merely delays execution of predetermined instructions until the status register contains specified status information.

In fact, according to the Summary of the Invention of Devins, the system is directed to allowing a programmer to code graphics applications to execute graphics operations without initiating host processor hardware interrupt handling routines, by monitoring the status indicator in the graphics accelerator and issuing hardware instructions in the captured programs based on the status information in the indicator. (See Devins, column 2, lines 55-61). That is, after a program is captured, instructions therein can cause the program to wait for a hardware event in the status register before executing program instructions. (See, for example, Devins, Fig. 4 item 300).

Column 6, lines 42-53, of Devins shows a captured program executing by continuously polling for a triggering event in a hardware status register. When the polling shows that the triggering event has occurred, then the programmed operations are executed. (See also column 8, lines 5-11, of Devins). In other words, the programmed operations are predetermined (since the program is already chosen) and merely wait for a triggering event before executing; but there is no discussion of automatically starting a job, *determined based on the recognized predetermined operation status*, as recited in independent claim 1, for example. That is, Devins does not teach or suggest determining which job to automatically start, based on a recognized status.

Further, in the Response to Arguments, on page 7 of the final Action mailed April 18, 2006, the Examiner states "Applicant argued that Devins does not teach automatically starting a job based on the recognized predetermined operation status." In response, the Examiner reiterates the reference to column 3, line 58, to column 4, line 7, of Devins as disclosing this feature.

However, independent claim 1, for example, does not merely recite automatically starting a job based on the recognized predetermined operation status, as the Examiner suggests. In contrast, claim 1, for example, recites "automatically starting a job, *determined* based on the recognized operation status."

That is, according to the present invention, the job to be automatically started is determined based on the operation status of the computer system, which is recognized based on whether a file corresponding to the predetermined operation status exists within the memory section.

In order to illustrate operations of the present invention, assume, as an example, that in a first operation state of a computer system both jobs 1 and 2 are executable; in a second operation state, job 1 is executable but job 2 is not; in a third operation state, job 1 is not executable but job 2 is executable; in a fourth operation state both jobs 1 and 2 are not executable. That is, according to independent claim 1, operation states of a computer system are recognized based on the existence of files, where these files are prepared or deleted according to changes in an operation state of the computer. The executable job is determined and automatically started is in response to the operation state of the computer system.

On the other hand, Devins substantially corresponds to the system described in the "Background Art" section of the present application, in which a program for recognizing an operation status of the system is prepared for each job within a program that executes the job based on a flag showing the operation status of the system. (See page 2, lines 23-27, of the

present application).

Furthermore, Devins does not disclose the use of various operation statuses as defined by the present invention recited in independent claim 1. That is, Devins does not teach or suggest "recognizing a predetermined operation status of the computer system, depending on whether a file corresponding to the predetermined operation status exists within the memory section or not."

According to the Abstract in Devins, status information relates to a plurality of graphics operations performed by a graphics accelerator, and does not relate to operation statuses of a computer system, as recited in independent claim 1, for example. The Examiner notes, on page 3 of the final Action dated April 18, 2006, that Devins does not explicitly teach controlling the operating of the operating system (within a computer system), but states that the system of Devins is implemented within an operating system. However, it is respectfully submitted that the Examiner has not cited prior art that teaches recognizing the operation status "of the computer system", as recited in independent claim 1. In contrast, the Examiner has merely cited a portion of Devins that discusses operations performed by a graphics accelerator, which is not synonymous to the computer system of independent claim 1.

Therefore, it is respectfully submitted that independent claim 1 patentably distinguishes over the cited reference. Independent claims 1, 16 and 17 recite features similar to those described above for independent claim 1. Thus, it is further submitted that independent claims 1, 16 and 17 patentably distinguish over the prior art for at least the reasons provided herein for independent claim 1. Dependent claims 2, 3, 6-15 and 18 inherit the patentability of their respective base claims and, therefore, it is submitted that the pending dependent claims also patentably distinguish over the cited art.

Group B: Claims 4 and 5

Dependent claims 4 and 5 inherit the patentable features of independent claim 1 and, thus, patentably distinguish over the prior art for the reasons set forth above. However, dependent claims 4 and 5 further recite, "the starting of the predetermined job is determined based on whether a plurality of the files exist or not within the memory section." (See, for example, page 4, lines 3-8, and page 8, lines 16-18, and Fig. 1, reference numeral S12, of the present application).

In rejecting these features, the Examiner cites column 3, line 58, to column 4, line 7, of Devins, which merely states that a device driver 15 including a capture routine 50 and I/O directives causes the DLP 25 to monitor the status of register 100 in the accelerator 30. The

device driver 15 simply delays execution of predetermined instructions until the status register contains specified status information.

However, the files, according to embodiments of the present invention, are prepared or deleted according to various operation statuses of the computer system, and the prepared files are stored in the memory. According to dependent claims 4 and 5, when a plurality of such files exists, the predetermined job is started.

It appears that the Examiner equates storing the executable instructions of Devins in the memory ready to be executed, to the plurality of files, according to claims 4 and 5. However, the executable instructions of Devins (which the Examiner also equates to the automatically-started job of the present invention), should not be considered synonymous to a plurality of the files, according to embodiments of the present invention. As stated above, the files of dependent claims 4 and 5 refer to the files prepared or deleted according to various operation statuses of the computer system and stored in the memory of the computer system. In fact, the executable instructions of Devins are not prepared according to an operation status of any computer system, but are merely predetermined and awaiting execution when appropriate status information is stored in the status register.

Therefore, it is respectfully submitted that Devins does not teach or suggest starting a job based on whether the plurality of files exist or not within the memory, as recited in dependent claims 4 and 5. As a result, it is further submitted that the rejections of claims 4 and 5 should be reversed.

Conclusion

In summary, Applicants submit that claims 1-18 patentably distinguish over the prior art. Accordingly, Applicants respectfully request reversal of the Examiner's rejections.

The Commissioner is authorized to charge any Appeal Brief fee or Petition for Extension of Time fee for underpayment, or credit any overpayment, to Deposit Account No. 19-3935.

Respectfully submitted,

STAAS & HALSEY LLP

Date: February 20, 2007

By: Michael P. Stanley

Michael P. Stanley
Registration No. 58,523

1201 New York Avenue, N.W., 7th Floor
Washington, D.C. 20005
(202) 434-1500
Facsimile: (202) 434-1501

VIII. CLAIMS APPENDIX

1. A method of controlling operation of an operating system in a computer system, the method comprising:

preparing or deleting various kinds of files that show various operation statuses of the computer system in which an operation status represents what process is currently under execution, according to changes in the operation status, and storing the prepared files in a memory section within the computer system;

recognizing a predetermined operation status of the computer system, depending on whether a file corresponding to the predetermined operation status exists within the memory section or not; and

controlling the operation of the operating system in accordance with a result of the recognition, thereby automatically starting a job, determined based on the recognized operation status, that can be executed in the operation status of the system after the operation status has been recognized.

2. The method of controlling the operation of an operating system in a computer system according to claim 1, wherein:

the control of the operation of the operating system is for starting a predetermined job.

3. The method of controlling the operation of an operating system in a computer system according to claim 2, wherein:

the predetermined job consists of a plurality of programs.

4. The method of controlling the operation of an operating system in a computer system according to claim 2, wherein:

the starting of the predetermined job is determined based on whether a plurality of the files exist or not within the memory section.

5. The method of controlling the operation of an operating system in a computer system according to claim 3, wherein:

the starting of the predetermined job is determined based on whether a plurality of the files exist or not within the memory section.

6. The method of controlling the operation of an operating system in a computer system according to claim 1, wherein:

each of the files is provided with an alias, and the operation status of the computer system is recognized based on the alias.

7. The method of controlling the operation of an operating system in a computer system according to claim 2, wherein:

each of the files is provided with an alias, and the operation status of the computer system is recognized based on the alias.

8. The method of controlling the operation of an operating system in a computer system according to claim 3, wherein:

each of the files is provided with an alias, and the operation status of the computer system is recognized based on the alias.

9. The method of controlling the operation of an operating system in a computer system according to claim 4, wherein:

each of the files is provided with an alias, and the operation status of the computer system is recognized based on the alias.

10. The method of controlling the operation of an operating system in a computer system according to claim 5, wherein:

each of the files is provided with an alias, and the operation status of the computer system is recognized based on the alias.

11. The method of controlling the operation of an operating system in a computer system according to claim 6, the method further comprising:

changing the operation status of the computer system based on starting of the predetermined job; and

starting a second job according to the changed new operation status of the computer system.

12. The method of controlling the operation of an operating system in a computer system according to claim 7, the method further comprising:

changing the operation status of the computer system based on starting of the predetermined job; and

starting a second job according to the changed new operation status of the computer system.

13. The method of controlling the operation of an operating system in a computer system according to claim 8, the method further comprising:

changing the operation status of the computer system based on starting of the predetermined job; and

starting a second job according to the changed new operation status of the computer system.

14. The method of controlling the operation of an operating system in a computer system according to claim 9, the method further comprising:

changing the operation status of the computer system based on starting of the predetermined job; and

starting a second job according to the changed new operation status of the computer system.

15. The method of controlling the operation of an operating system in a computer system according to claim 10, the method further comprising:

changing the operation status of the computer system based on starting of the predetermined job; and

starting a second job according to the changed new operation status of the computer system.

16. A computer-readable recording medium that has been recorded with a program for making a computer execute a method of controlling the operation of an operating system in a computer system, the recording medium being recorded with a program comprising:

preparing or deleting various kinds of files that show various operation statuses of the computer system in which an operation status represents what process is currently under execution, according to changes in the operation status, and storing the prepared files in a memory section within the computer system;

recognizing a predetermined operation status of the computer system, depending on

whether a file corresponding to the predetermined operation status exists within the memory section or not; and

controlling the operation of the operating system in accordance with a result of the recognition, thereby automatically starting the job, determined based on the recognized operation status, that can be executed in the operation status of the system after the operation status has been recognized.

17. A method of controlling the operation of an operating system in a computer system, the method comprising:

automatically recognizing an operation status of the computer system in which the operation status represents what process is currently under execution; and

automatically starting a job, determined based on the recognized operation status.

18. A method of controlling the operation of an operating system in a computer system according to claim 17, wherein:

the job is automatically executed in an operation status of the system after said operation status has been automatically recognized.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.